GlobalLogic®

A Hitachi Group Company

Java HandsOn Assessment

Contact Management System

- Title of the Project - Contact Management System
- Complexity - Medium
- Target Band - Band x / Band 1 / Band 2
- Downloadable Starter Code link: Link
- Skills: Java, Junit and Mockito, PostgreSQL.
- Time taken to complete: 3 Hours
- IDE: IntelliJ, Eclipse

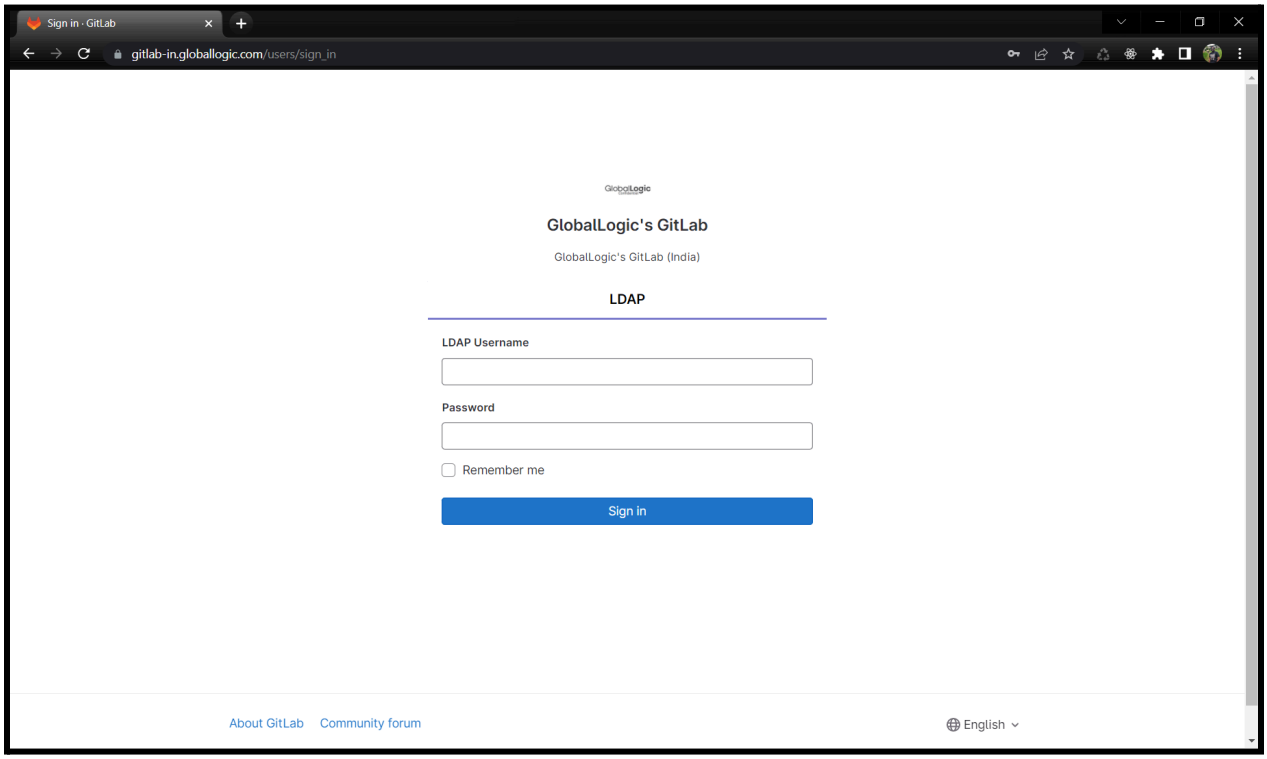
Technology	Topics
Java	OOPs, JDBC, Abstract Class and Interfaces, Exception Handling, Collections And Generics, Streams API, Lambda Expression, JUnit5 And Mockito, PostgreSQL.

Table of Contents

1. Problem Statement 6
2. DB Design 8
Contact 8
4.Implementation 9
Class / Method Description 9
5.Testing **13**
Instructions to Create and Execute the Project 13
Evaluation Metrics 14

GITLAB GENERAL INSTRUCTIONS

1. Connect to VPN to access GlobalLogic GITLAB
2. Open the URL in the browser - https://gitlab-in.globallogic.com/users/sign_in

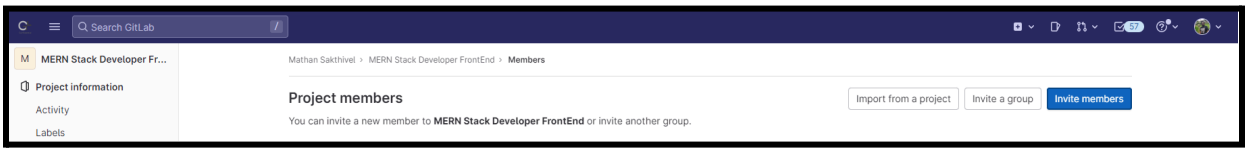


3. Login using GlobalLogic credentials.
4. Find a project to fork: Navigate to the project that you want to fork on GitLab. You can find projects by browsing the public repositories or by searching for a specific project.
5. Fork the project: Once you have found the project you want to fork, click the "Fork" button in the top right corner of the project page. This will create a copy of the project in your GitLab account.

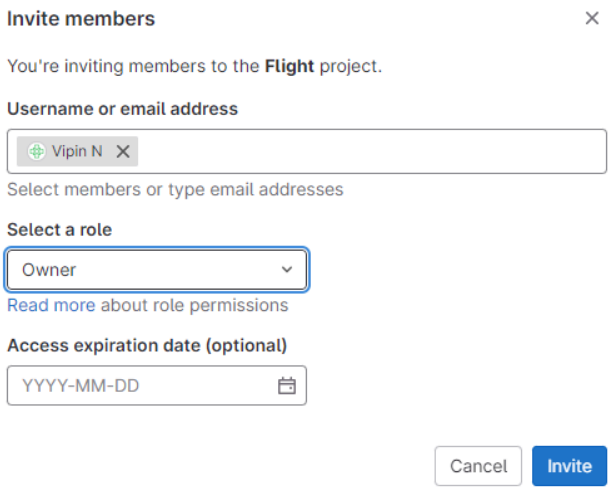
6. Clone the forked project: Once you have forked the project, clone the project to your local machine using Git. To do this, open a terminal window and navigate to the directory where you want to clone the project. Then, run the following command:
   git clone [URL of the forked project]
7. Make changes to the project: Once you have cloned the project to your local machine, make any necessary changes to the project files using your preferred code editor.
8. Commit your changes: Once you have made changes to the project files, stage the changes and commit them to your local repository using the following commands:
   git add.
   git commit -m "Your commit message here"

9. Push your changes to GitLab: Once you have committed your changes to your local repository, push the changes to GitLab using the following command:
   git push origin [name of your branch]
10. The forked project will have the Private Access so explicitly add the users in the project members.

    Click on Invite Members



Provide access to the Educators:
Username: Ex: minakshi.sharma@globallogic.com and vipin.n@globallogic.com
Role: Owner



SYSTEM REQUIREMENTS

1. Eclipse/STS/IntelliJ IDEA
2. PostgreSQL
3. Java 17 or above

Contact Management System

1. Problem Statement

The Contact Management System is a Java Application that simplifies contact organization and retrieval. It allows users to add and retrieve contacts effortlessly, including essential details like names, email addresses, and phone numbers. The system aims to streamline contact management, providing a user friendly solution for efficient organization and easy access to important contacts.

You are working as a Java developer in the Contact Management System project team.Your task is to create methods that allow users to add new contacts and retrieve the contact list. The system should validate the required contact details, such as name, email address, and phone number, and store the contacts securely. Additionally, the system should support sorting the contact list based on specified criteria..

User Stories

User Story	Description	Acceptance Criteria
US01	As a user, I want to be able to add a new contact to the system so that I can keep track of my contacts.	• I should be able to provide the name, email, and phone number of the contact. • The system should validate the provided details. • The system should inform me if the contact is added successfully. • The system should inform me if there was an error while adding the contact.

US02	As a user, I want to be able to retrieve the list of all my contacts so that I can view and manage them.	1. I should be able to request the list of all contacts. 2. The system should display the list of all contacts. 3. The system should inform me if there was an error while retrieving the contacts.
US03	As a user, I want to be able to sort my contact list based on different criteria so that I can easily find the contact I am looking for.	● I should be able to specify the criteria for sorting (name, email, or phone number). ● The system should sort the contact list based on the specified criteria. ● The system should display the sorted list of contacts. ● The system should inform me if there was an error while sorting the contacts.
US04	As an admin, I want to be able to view the details of a specific contact so that I can verify or review the contact information.	● I should be able to provide the id of the contact. ● The system should display the details of the specified contact. ● The system should inform me if there was an error while retrieving the contact details.
US05	As an admin, I want to be able to import multiple contacts at once from a data stream and store them in the database.	● Given a stream of contact data, when I invoke the import function, then it should process each item in the stream. ● The import function should transform each item into a Contact object. Each item in the stream is a string that

		contains contact information in a specific format, for example, “contact_id,name,email,phoneNumber". ● The import function should then store each Contact object into the database.

2. DB Design

Contact

Column Name	Description	Data Type	Not Null
contact_id	This is the primary key of the Contacts table. It is an integer value that uniquely identifies each contact in the table. In a real-world application, this would typically be auto-incremented.	INT	NOTNULL
name	This field stores the name of the contact.	VARCHAR(255)	NOTNULL
email	This field stores the email address of the contact.	VARCHAR(255)	NOT NULL
phoneNumber	This field stores the phone number of the contact.	VARCHAR(255)	NOT NULL
Primary Key(contact_id)			

4.Implementation

Class / Method Description

This is a standalone Java application where a menu-driven interface needs to be developed to execute the following functionalities.

Note:- The above UserStories are converted into equivalent methods and the descriptions of the service methods provided below need to be implemented.

Entity Class	Service Methods	Description
Contact(contact_id,name,email,phoneNumber) Given Contact class in the starter code convert it into Java 17 Records to store the Contact details		
	addContact(String name,String email,String phoneNumber)	Add a new Contact
	getContacts()	Returns a List of contacts
	sortContact(String criteria) Use Java8 Streams API for sorting	Sorts the contacts based on criteria
	getContactByID(int id) Use Java8 Streams API for searching	Retrieves a specific contact by ID

	importContacts(Stream< String> contactData) Use Java8 Streams to import the contactData	Given a list of contact data, the method will import multiple contacts from this data stream and store them in the database.
Primary Key(contact_id)		

Output screens

1) Adding a New Contact Information

i) If the entered contact details are invalid

ii) if the contact details are valid the contact details are successfully added in the database table.

```
Enter your choice: 1
Enter name: Mona
Enter email: mona@gmail.com
Enter phone: 9989898801
Contact added successfully!
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
Enter your choice:
```

2. Display List of contacts

```
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
Enter your choice: 2
Contact List:
Contact[contact_id=2000, name=Amit, email=amit@gmail.com, phoneNumber=9088877777]
Contact[contact_id=2001, name=Sonal, email=sonal@ymail.com, phoneNumber=7798798989]
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
```

3. Sort contacts List based on the entered criteria

```
Enter your choice: 3
Sort contact List:
Enter criteria: name
Contact[contact_id=2000, name=Amit, email=amit@gmail.com, phoneNumber=9088877777]
Contact[contact_id=2001, name=Sonal, email=sonal@ymail.com, phoneNumber=7798798989]
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
```

4.Get contacts by Id

```
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
Enter your choice: 4
Enter contact ID: 2000
Contact: Contact[contact_id=2000, name=Amit, email=amit@gmail.com, phoneNumber=9088877777]
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
```

5.Import Contacts from data streams

```
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
Enter your choice: 5
Contacts imported successfully!
1. Add contact
2. Display contact List
3. Sort contact List
4. Get contact by ID
5. Import contacts
6. Exit
```

6.Exiting the application

```
Enter your choice: 6
Exiting...
```

5.Testing

Write a Test case for the addContact() Method using Junit 5 :

- Use JUnit annotations to denote setup and test methods (@BeforeEach and @Test).
    - The setUp method initializes a Baggage instance with baggage details.

- Create a unit test method named `testAddContact()`. This method should verify that the `addContact()` method of a mock service (presumably `contactService`) was called exactly once, and that it was called with the correct argument, which is a `contact` object. The `verify` method from the Mockito testing framework is suggested for this purpose.

Instructions to Create and Execute the Project

- Please download the code structure from the provided GitLab link. The project structure can be found under the 'com.gl.app' package and its subpackages.
- Please design the necessary database schema to store Contact information in the database table.
- Create a Menu driven Java Project which can add Contact Details
- Use PostgreSQL as the database management system.
- In addition to implementing the functionalities mentioned above.
    - Read Input from the keyboard using Scanner class.
    - You should include code for establishing a database connection, executing queries, and handling exceptions.
    - Use Java 8 Lambda Expressions,Streams API.
    - Field Validations(Acceptance Criteria)
        - Validate the Contact details(name,email).
        - contact_id should have unique values in the Contact table.
        - Handle and display errors gracefully to the user in case of invalid inputs or other issues.

Evaluation Metrics

Sl. No.	Assessing Parameter	Marks
1	User Stories 5* 7 marks	35 marks
2	Usage of Streams and Records	5 marks
3	Testing Module	5
4	Exception Handling and following best practices	5
	Total Marks	50 Marks